NASA-CR-193411

JJM
SYSTEMS INC

C931032-U-2R08

AN EXPERT SYSTEM SHELL FOR INFERRING VEGETATION CHARACTERISTICS - PROTOTYPE HELP SYSTEM (TASK I)

NAS5-30127

July 1993

Prepared for:

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, MD 20771

Prepared by:

JJM Systems, Inc.
1225 Jefferson Davis Hwy., Suite 412
Arlington, VA 22202

(NASA-CR-193411) AN EXPERT SYSTEM SHELL FOR INFERRING VEGETATION CHARACTERISTICS: PROTOTYPE HELP SYSTEM (TASK 1) (JJM Systems) 24 p

N94-11236

Unclas

G3/43 0176695

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS

KEE	Knowledge Engineering Environment
VEG	VEGetation Workbench

SECTION 1.0

INTRODUCTION

The NASA VEGetation Workbench (VEG) infers vegetation characteristics from reflectance data. VEG was developed using the Intellicorp, Knowledge Engineering Environment (KEE). KEE is a mature development platform that supports a number of well-engineered components including inference engines, windows, graphics tools, objects and inheritance, procedural attachments and other support needed for prototyping expert systems using object-oriented programming.

An extensive, window-oriented interface system was constructed for VEG using the KEE graphics package called "ActiveImages." This interface provides a variety of screens to enhance dialogue between the scientist and the system. The interface is a key feature of this system. It was designed to focus the scientist on the appropriate level of organization to carry out scientific work without attention to "housekeeping" functions. The interface allows the scientist to run VEG and select options at all stages of the run by clicking the mouse over the appropriate menu option. The interface further allows the scientist to focus on the data and the functions performed by VEG as it abstracts away most of the underlying, detailed complexity of the VEG system.

A prototype Help System has been designed and implemented. The Help System allows the scientist to get more information about each screen in the VEG interface. It was designed to help the new user of VEG to learn how to operate the system. The Help System is stored in separate files from the VEG knowledge base and it is loaded only when needed. An interface that allows the scientist to add and modify help messages has also been integrated into the "Administration" part of the VEG system. This enables the scientist to evolve the Help System over time.

Task I of this project required the design and implementation of a prototype Help System. This task has been completed. The code for the Lisp methods used in this task is included in Appendix A. A Sun cartridge tape containing the Lisp methods and the current version of VEG, including the Help System has been delivered to the NASA GSFC technical representative.

SECTION 2.0

DESCRIPTION OF THE HELP SYSTEM

The storage of help messages in the VEG system and the method of operation of the Help System are described in this section.

2.1 STORAGE OF HELP MESSAGES IN VEG

The screens in the VEG interface were built using the KEE ActiveImages package. The attributes of each screen are stored in KEE units known as "Viewports" and a number of other units that hold the attributes of the screen subwindows. The Help System provides help for each VEG screen. When the Help System is loaded, an extra slot called "HELP" is created in each viewport unit. This slot holds the help message for the screen to which the viewport unit refers.

The unit HELP.SYSTEM has been created in VEG. This unit holds the slots required by the Help System. The slots in this unit are shown in Figure 2-1. The HELP.LOADED slot is initialized with the value NIL to indicate that the Help system is not loaded. The scientist must explicitly load the Help System. When the Help System is loaded, the value of HELP.LOADED is changed to T for true. The rationale for this approach is that the Help System will be used less as the scientist gains proficiency with the VEG system. Therefore, when possible, the VEG system avoids the overhead of having the Help System loaded.

HELP.LOADED
MESSAGE
OPTIONS

Figure 2-1
Slots in the Unit HELP.SYSTEM

The VEG system uses four separate knowledge bases: VEG, LEARN, AZIMUTH.PLOT and POLAR.PLOT. Each knowledge base contains at least one viewport. The help messages for the four knowledge bases in the VEG system are stored in the files "help-messages-veg," "help-messages-learn," "help-messages-azimuth," and "help-messages-polar," respectively. The files hold the viewport name and the appropriate help message for each viewport that has help available. When the Help System is loaded, the slot "HELP" is first added to the units that are parent units of the viewport units so that the HELP slot is inherited by each viewport unit. The help messages for the VEG knowledge base are then read from the "help-messages-veg" file and stored in the newly created HELP slots. Checks are then made to determine whether any of the other VEG knowledge bases have been loaded. If any additional knowledge bases have been loaded, then the help messages for these knowledge bases are also read from the files and stored in the knowledge bases. The value of the HELP.LOADED slot of the unit HELP.SYSTEM is then changed to T. If any of the knowledge bases LEARN, AZIMUTH.PLOT or POLAR.PLOT are subsequently loaded, the help messages for the additional knowledge base are read from the appropriate help message file and stored in the knowledge base immediately after it has been loaded.

2.2 THE OPERATION OF THE HELP SYSTEM

The user can activate the Help System at any time when VEG is loaded, by left clicking on the HELP.SYSTEM option in the Tool Box Menu as shown in Figure 2-2. If HELP.SYSTEM is moused in the Tool Box Menu, a Lisp method checks the value of the HELP.LOADED slot of the HELP unit and loads the Help System if it has not been loaded. The Help Screen is then opened as shown in Figure 2-3. The ActiveImages in this screen are attached to slots in the HELP.SYSTEM unit. The user is prompted to mouse on the window he/she needs help with, and the cursor changes to a cross shape. When the user left-clicks on a window, the Help System identifies the window that has been moused on. If the window is a KEE viewport, the help slot of the viewport unit corresponding to the window is examined. If a help message is found, the help message is put into the MESSAGE slot and consequently displayed on the Help System Screen, as shown in Figure 2-3. If no help for the window is available, the message "Sorry, no help is available for this window" is displayed. If the user has selected a window that is not part of the VEG system, such as the KEE typescript window or the Open Windows Workspace, the message "Not a VEG window" is displayed on the Help System Screen.

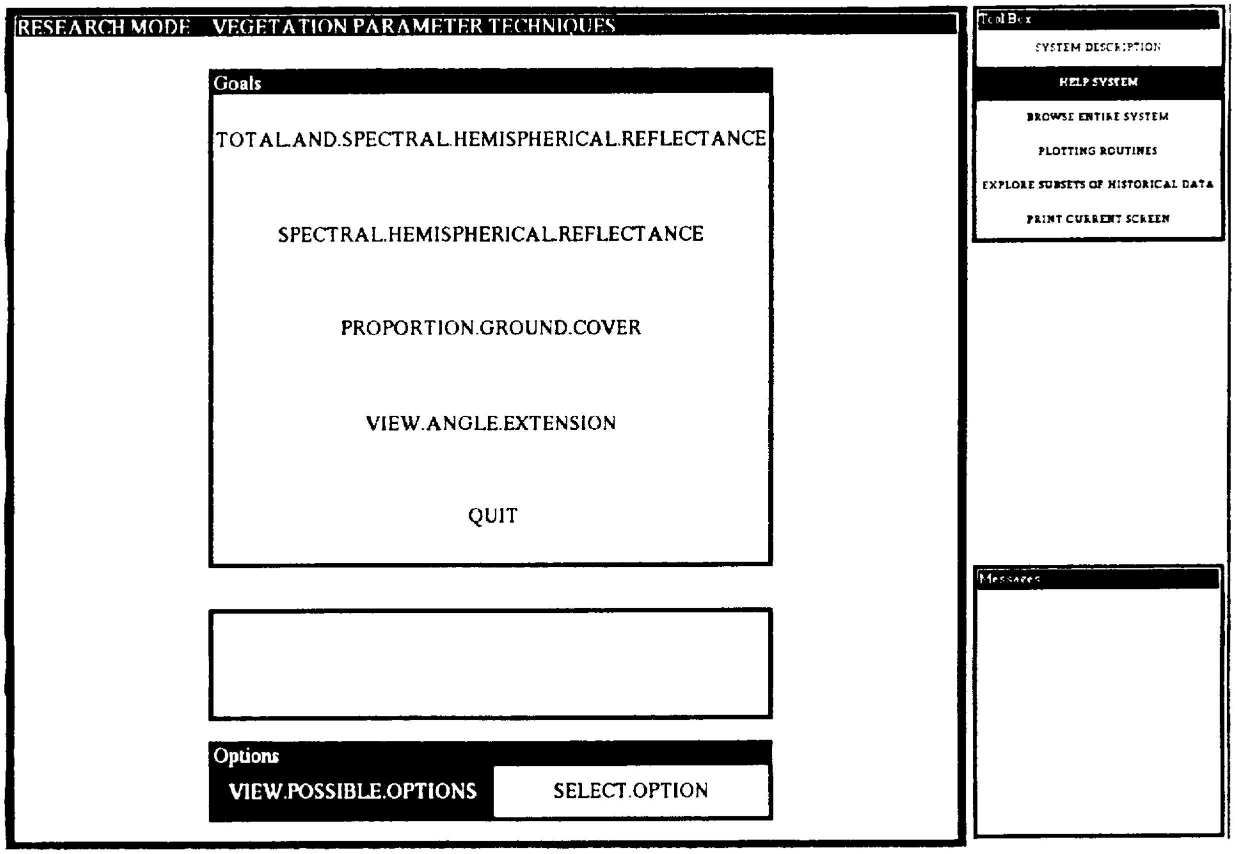
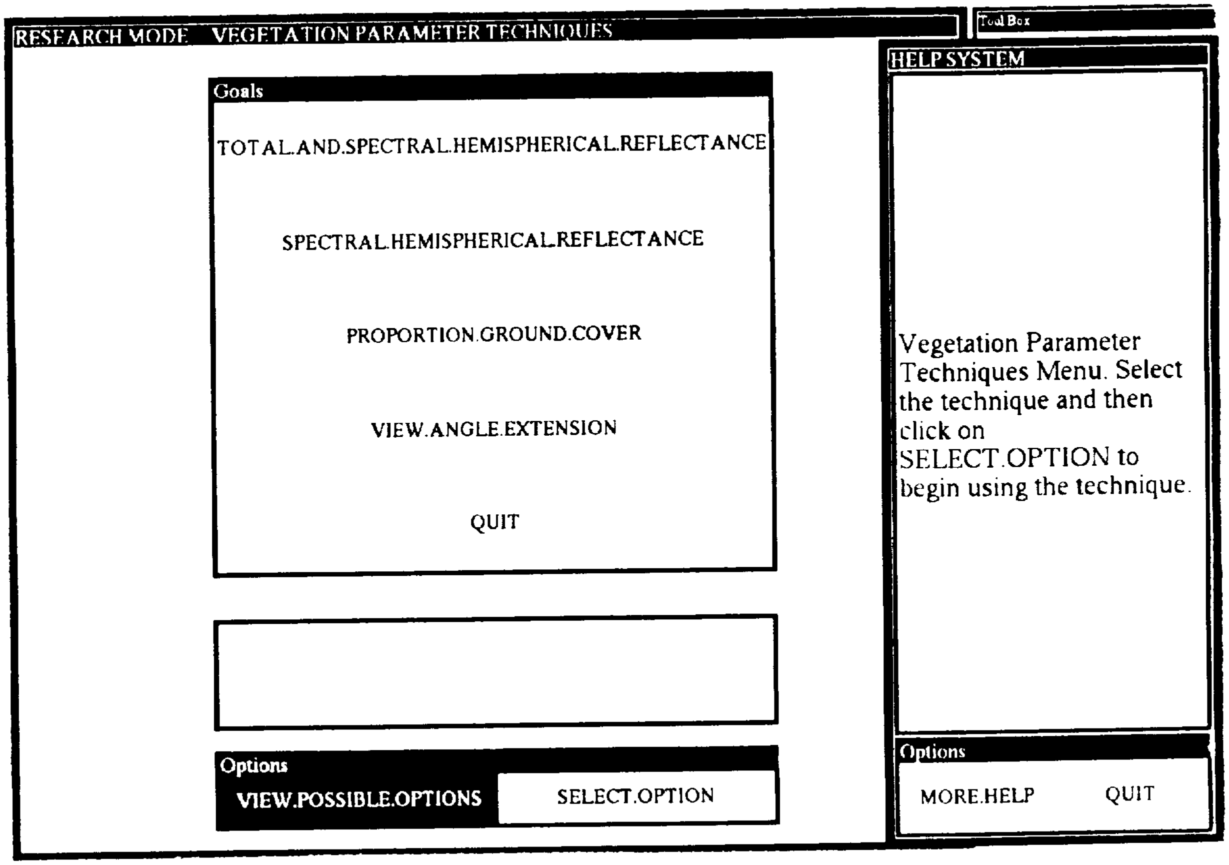


Figure 2-2
Using the Help System

The user can continue to use VEG while the Help System Screen is open, although one or more windows might be partially occluded by the Help System Screen. If the user left clicks on MORE.HELP, the cursor changes to a cross again and the user is prompted to select another screen for help. Once opened, the Help System Screen remains open until the user closes it by left clicking on the QUIT option at the bottom right of the screen.



Vegetation Parameter Techniques Menu. Select the technique and then click on SELECT.OPTION to begin using the technique.

Figure 2-3
The Help Screen

SECTION 3.0

ADDING AND MODIFYING HELP MESSAGES

An option that allows the scientist to add or modify help messages has been added to the Administration part of VEG. This allows the scientist to evolve the Help System interactively. It recognizes that help concepts will evolve and change as the scientist gains experience using various system functions. VEG is also extensible in certain ways which might require the addition and/or modification of help messages.

If the user left clicks on RUN.VEG and then selects ADMINISTRATION from the Processing Mode menu, the Administration Screen is opened as shown in Figure 3-1. The option CHANGE.HELP.MESSAGES has been added to the Administration Menu for handling message changes. When the user selects CHANGE.HELP.MESSAGES, the Administration Screen is closed and the Add/Change Help screen is opened, as shown in Figure 3-2. The user is prompted to navigate through the VEG system until the screen is reached that is to be associated with the new or modified help message. When the screen level is located, the user left clicks on MODIFY.HELP in the option section of the Add/Change Help window. When this is done, the cursor changes to a cross and the user is prompted to left click on the screen for which help is to be added or modified. When the cursor is brought over the appropriate window and left clicked, the New Help Message window is opened. This is shown in Figure 3-3. The Add/Change Help Window allows the user to iterate the process of modifying and saving help messages. The New Help Message window displays the value "Unknown" if this is a new help message as shown in Figure 3-3. If the user is modifying an existing message, the current help message is displayed in the window as shown in Figure 3-4. Once the new message is typed into the window, as shown in Figure 3-5, or the existing message has been modified, left clicking on DONE will close the window. This process can be continued by left clicking on MODIFY.HELP in the options section of the Add/Change Help window. The user can choose to click on SAVE.HELP at any time in order to save any changes made to the Help System. Finally, when the user is done adding and modifying the Help System, left clicking on QUIT in the options section of the Add/Change Help window will close the window and return control to the Administration window. If messages are not saved, they will be lost when VEG application is exited.

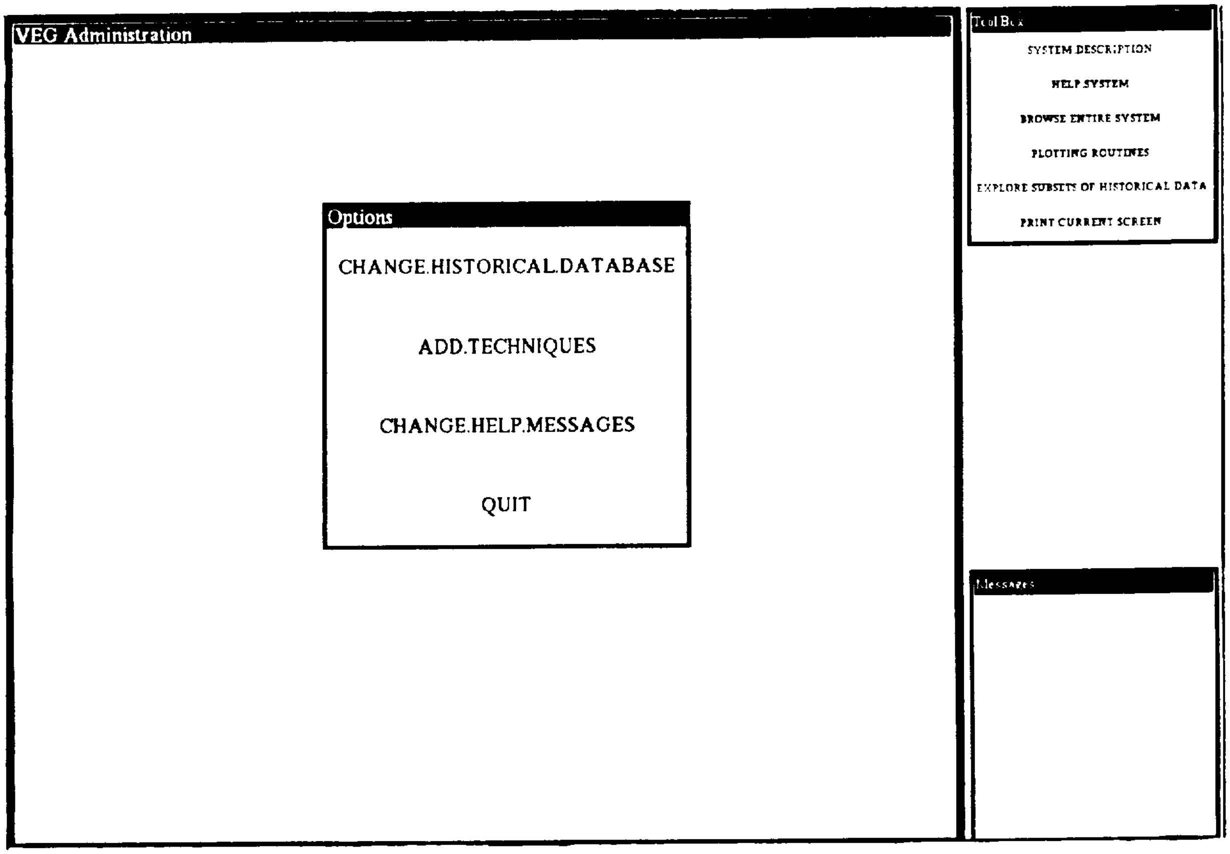
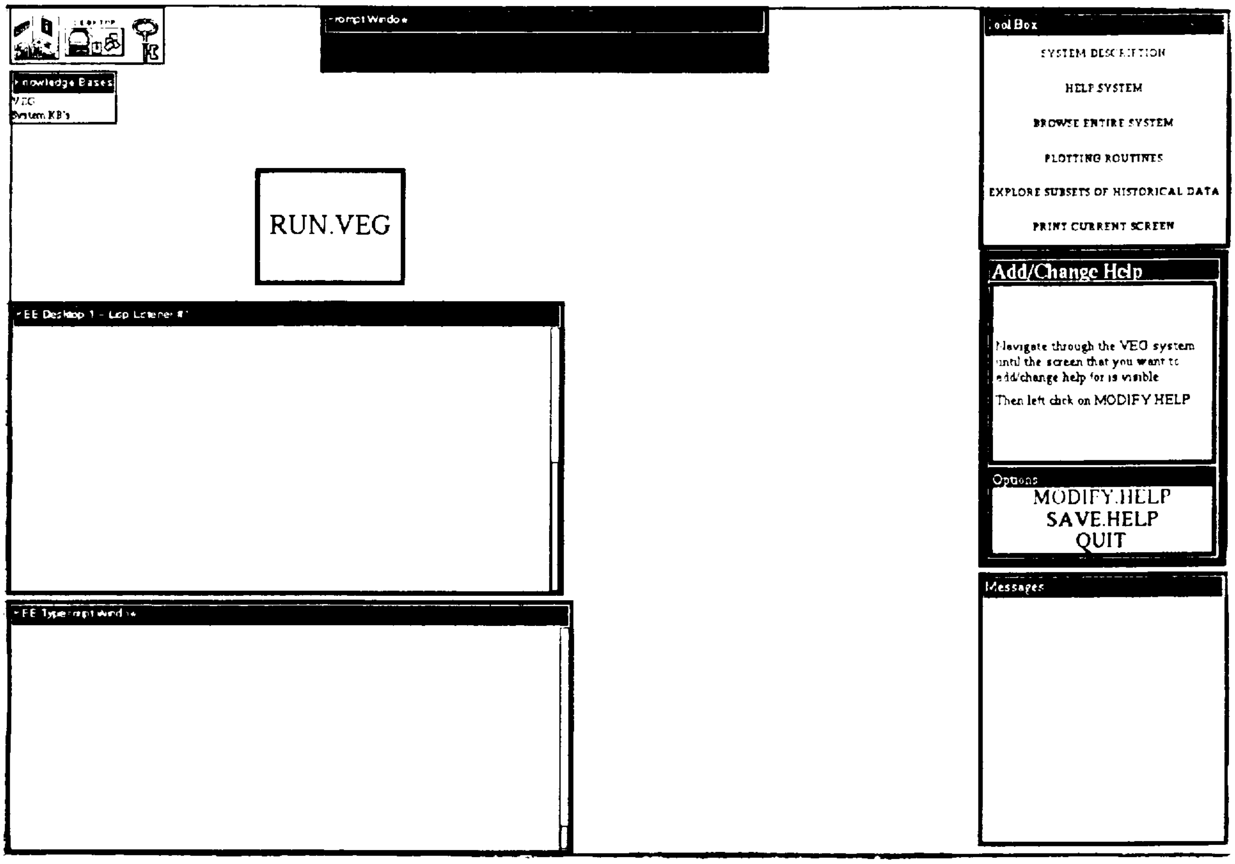


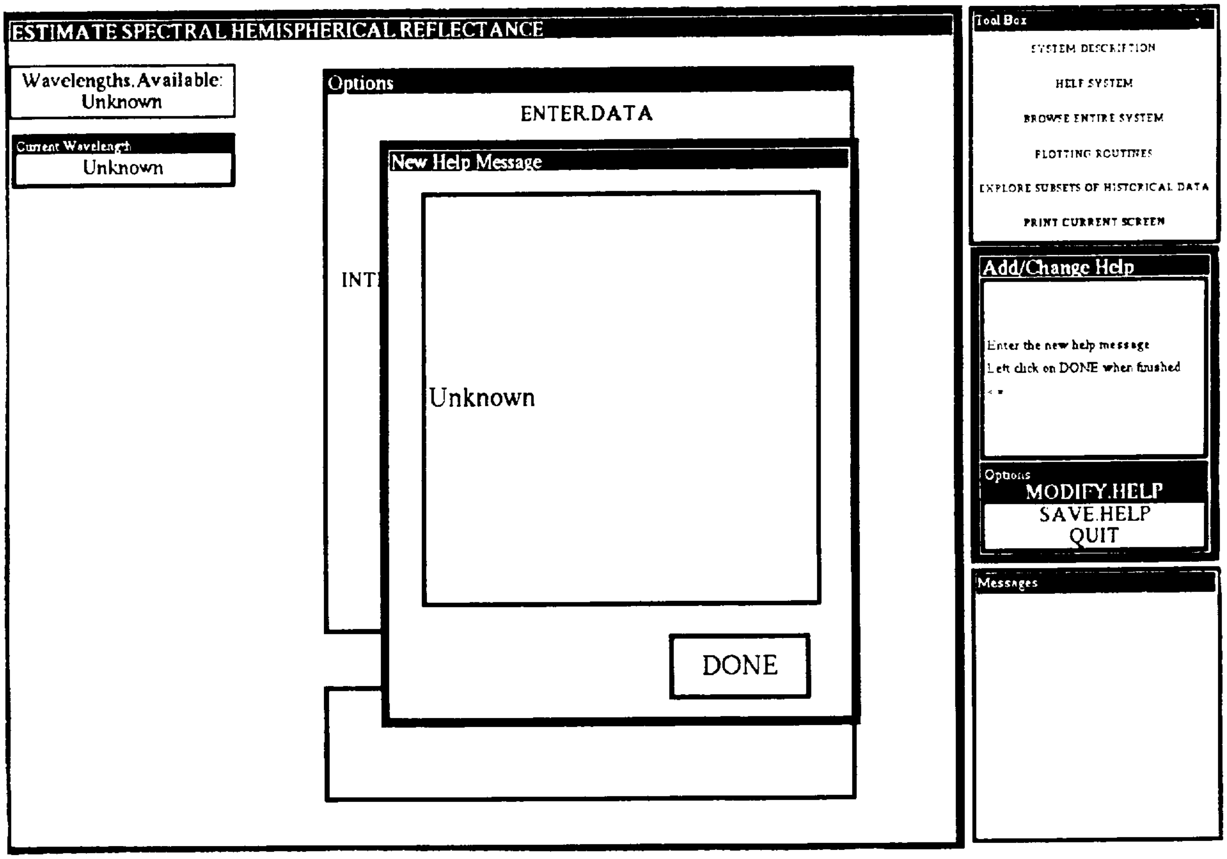
Figure 3-1
VEG Administration Screen with CHANGE.HELP.MESSAGES



Navigate through the VEG system until the screen that you want to add/change help for is visible

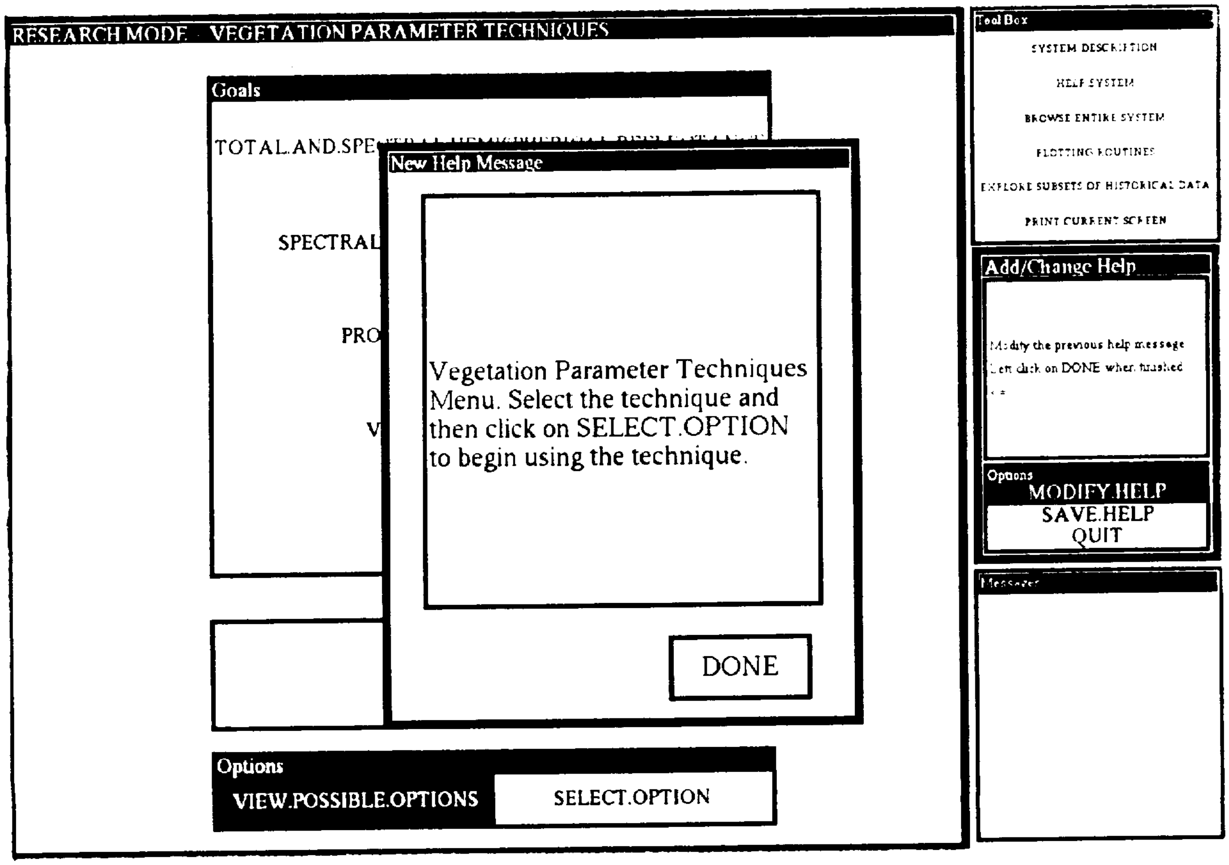
Then left click on MODIFY HELP

Figure 3-2
Add/Change Help Screen with Options



Enter the new help message
Left click on DONE when finished
< >

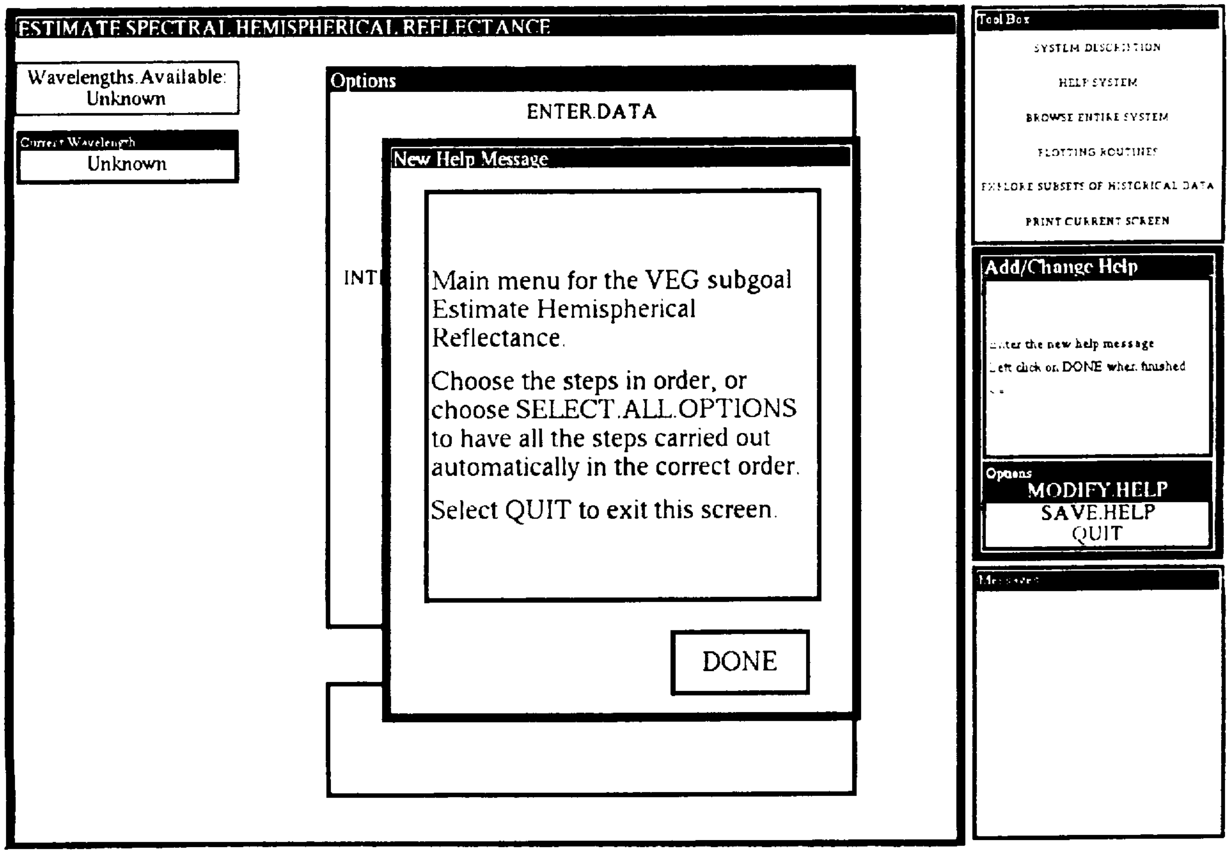
Figure 3-3
New Help Message Screen before a Message is Added



Vegetation Parameter Techniques Menu. Select the technique and then click on SELECT.OPTION to begin using the technique.

Modify the previous help message
Left click on DONE when finished

Figure 3-4
New Help Message Screen Before an Existing Message is Modified



Main menu for the VEG subgoal Estimate Hemispherical Reflectance.

Choose the steps in order, or choose SELECT.ALL.OPTIONS to have all the steps carried out automatically in the correct order.

Select QUIT to exit this screen.

Enter the new help message
Left click on DONE when finished

Figure 3-5
New Help Message Screen with New Message

The Help System works equally well if multiple knowledge bases (application components) are loaded. The Help System automatically determines the knowledge base with which a window is associated. When a save is initiated, the help message is saved in a file using a name that includes the knowledge base name. For example, if a help message was added to a Learning System window, then an ASCII file named "help-messages-learn" would be created (if it did not already exist) and the message would be saved in the new file or appended to the existing file. Currently, help files are saved as text files. The help files can be inspected outside of the VEG system. It is possible to modify the help messages in the files using an editor. However, this is not recommended. Any editing should be done with great care. The help messages are stored in the files together with the object identifiers for the screens to which the messages apply. Changing the organization of the help message files would cause errors which would prevent the Help System being loaded. When help messages are added or modified and then saved using the VEG interface, they are automatically saved in the correct format.

It should be noted that with minimal effort, the new help message entry window could be replaced by an editor window tied to the editor favored by the user. This would allow more extensive editing capabilities than are presently available.

SECTION 4.0

TESTING AND RESULTS

The following capabilities of the VEG Help System and the Add/Modify Help option were tested:

- Test 1 - Navigate through an empty Help System.
- Test 2 - Add help messages to windows. Save help messages and inspect saved files.
- Test 3 - Navigate through the Help System and read previously saved help messages.
- Test 4 - Modify help messages in existing windows.
- Test 5 - Add help messages to multiple knowledge bases.

All the tests were successful, showing that the system was working correctly. The tests are described in detail in this section.

4.1 TEST 1

Test 1 simply navigated through the Help System before any help messages were added. This was done with one and two knowledge bases active at the same time. Application windows as well as KEE and OPENWIN application icons were tested. After left clicking on HELP.SYSTEM in the Tool Box menu, the message "Click on the window you need help with" appeared. The cursor became a cross shape. Placing the cross over the window of interest and then left clicking produced the message, "Sorry, no help currently available for this window" if the window was an application window, or "Not a VEG window" if the window was a KEE or OPENWIN window. No errors occurred. When an application window was clicked regardless of the knowledge base, the system behaved as expected. This test demonstrated the basic functionality of the HELP system for navigating through an application.

4.2 TEST 2

Test 2 activated the ADMINISTRATION window and then activated the CHANGE.HELP.MESSAGES window. The tester then navigated to the Automatic Mode Screen and clicked left on the MODIFY.HELP option in the Add/Change Help Screen. The cursor changed to a cross and the tester left clicked on the Automatic Mode Screen to indicate that help for this screen was to be added. A New Help Message window appeared and a new message was added. DONE was left clicked in the New Help Message window to indicate the message was complete. The procedure was repeated using several different windows at different levels in the VEG application. Each time a new message was added, MODIFY.HELP was left clicked to reiterate the process. Once messages had been added to different screens, the Help System was again invoked and the help messages successfully displayed. Finally, the SAVE.HELP option was activated to save the help messages that had been entered so far. A file called "*help-messages-veg*" was created and contained the help messages that had been entered.

4.3 TEST 3

The VEG system was exited and re-loaded. The HELP.SYSTEM option was selected from the Tool Box Menu. The Help System was loaded and the messages that had been added in Test 2 were successfully accessed through the Help System.

4.4 TEST 4

In Test 4, the ADMINISTRATION window was again activated, and the CHANGE.HELP.MESSAGES window activated. The tester then navigated to a window, for which a help message existed, and clicked left on the MODIFY.HELP option. The cursor changed to a cross and the tester left clicked on the window whose help message was to be modified. A New Help Message window appeared and the current message was displayed. The message was changed by writing a new message. DONE was left clicked in the New Help Message window to indicate the modification was complete. This was repeated using several different windows at different levels in the VEG application. Each time a message was modified, MODIFY.HELP was left clicked to reiterate the process. Once this process was completed, the Help System was again tested and the help messages successfully displayed. Finally, the SAVE.HELP option was activated to append the modified help messages to the existing file. The file was inspected and properly saved. The VEG system was exited and reloaded. The Help System was loaded and the messages were successfully accessed through the Help System.

4.5 TEST 5

Test 5 replicated the elements of tests 1 through 4 with multiple knowledge bases (modules) loaded in the VEG system. In addition to the VEG core, the AZIMUTH PLOT, POLAR PLOT and LEARN knowledge bases were loaded. Then tests 1 through 4 were repeated using windows from the four components. The system again performed as expected. The VEG help file was properly updated and new help files called "*help-messages-azimuth*," "*help-messages-polar*," and "*help-messages-learn*" were created.

4.6 RESULTS

The test suite demonstrated the ability of the Help System to provide the range of behavior expected of the Help System prototype.

SECTION 5.0

CONCLUSIONS

The prototype Help System provides an interactive tool for adding help support to the VEG system. It was designed to enable the scientist to control and shape the help facility without bothering with the details of implementation. The Help System provides both a help system and a tool for developing new help messages and modifying existing help messages. File management and object management issues are transparent to the user. Currently, the editing facilities for message modification are minimal. The Help System was designed so that it would be simple to replace the current editing window with whatever editor (emacs, textedit, vi) the user might favor for adding or modifying messages.

Since the Help System may not be needed by an experienced user, it was configured so that it is loaded only when the user initially clicks on the Help System option in the Tool Box Menu. This minimizes the overhead for the VEG environment.

APPENDIX A

LISP CODE FOR THE PROTOTYPE HELP SYSTEM

```
;;; veg-methods7.lisp
;;;
;;; Created April 27, 1993
;;; Last Modified July 22, 1993

(in-package 'kee)

(defun start-help-system ()
"Starts the help system."
  (remove.all.values 'help.system 'options)
  (cond ((get.value 'help.system 'help.loaded)
         (put.value 'help.system 'message
           "Click on the window that you need help with")
         (unitmsg 'viewport-help.system.1 'open-panel!))
        (t (put.value 'help.system 'message "Loading help ........")
          (unitmsg 'viewport-help.system.1 'open-panel!)
          (load-help)
          (put.value 'help.system 'message
            "Click on the window that you need help with")))
  (show-text))

(defun load-help()
"Call the function to load help messages from the appropriate files into the
help slots of viewports."
   (add-help-slots-to-viewports)
   (add-help-messages "help-messages-veg")
   (when (kb.exists.p 'learn)
    (add-help-messages "help-messages-learn"))
   (when (kb.exists.p 'azimuthplot)
    (add-help-messages "help-messages-azimuth"))
   (when (kb.exists.p 'polarplot)
    (add-help-messages "help-messages-polar"))
   (put.value 'help.system 'help.loaded t))

(defun add-help-messages (file)
"Load help messages from a file into the help slots of viewports."
   (with-open-file (str file :direction :input :if-does-not-exist :nil)
     (when str
       (do ((win (read-file str)(read-file str)))
            ((null win) (values)) ; End of file
         (if (unit.exists.p win) ; Window is found
              (put.value win 'help (read-file str)) ; Read & store message
              (read-file str)))))    ; Read past unused message

(defun add-help-slots-to-viewports()
"Modify the viewport parent units in the ACTIVEIMAGES knowledge base in
preparation for storing the help messages in the slots of each viewport unit."
   (create.slot 'ai3-kb-viewports 'help 'member "")
   (add.value 'ai3-kb-viewports 'local.compact.unit.slotnames 'help)
   (create.slot 'ai3-unit-viewports 'help 'member "")
   (add.value 'ai3-unit-viewports 'local.compact.unit.slotnames 'help)
   (create.slot 'ai3-slot-viewports 'help 'member "")
   (add.value 'ai3-slot-viewports 'local.compact.unit.slotnames 'help))
```

```
(defun get-more-help ()
"Prompts the user to select the screen for additional help."
  (put.value 'help.system 'message
     "Click on the window that you need help with")
  (show-text))

(defun mouse-top-window ()
"This function allows the user to mouse directly on the window that represents
the object he needs help with."
   (let ((pos (get-position)))
     (window-stream-under-position pos)))

(defun unit-from-stream (window)
"Returns the name of the viewport corresponding to the window."
   (getf (kwin-plist window) 'viewport))

(defun show-text ()
"Returns the help message from the moused window."
   (let ((unit (unit-from-stream (mouse-top-window))))
     (put.value 'help.system 'message
           (if (not unit)
              "Not a VEG window"
              (let ((help (get.value unit 'help)))
                (if help
                      help
                      "Sorry, no help currently available for this window"))))))

;;; ---------------------------------------------------------------------
;;; Methods for Changing or Adding Help Messages
;;; ---------------------------------------------------------------------

(defun open-change-help-menu ()
"Opens the top screen for changing or adding help messages."
  (remove.all.values 'add.help 'options)
  (cond ((get.value 'help.system 'help.loaded)
          (put.value 'add.help 'message
"Navigate through the VEG system until the screen that you want to add/change help for is visible.
Then left click on MODIFY.HELP.")
          (unitmsg 'viewport-add.help.2 'open-panel!))
         (t (put.value 'add.help 'message "Loading help ........")
            (unitmsg 'viewport-add.help.2 'open-panel!)
            (load-help)
            (put.value 'add.help 'message
"Navigate through the VEG system until the screen that you want to add/change help for is visible.
Then left click on MODIFY.HELP.")))
  (remove.all.values 'workbench 'run.veg)
  (unitmsg 'viewport-run.veg-of-workbench.1 'open-panel!))

(defun modify-help ()
  (put.value 'add.help 'message
     "Left click on the window that you want to change the help on")
  (add-help))
```

```
(defun save-help ()
"Saves the modified help messages to the help file."
   (with-open-file (strv "help-messages-veg" :direction :output
                                   :if-does-not-exist :create :if-exists :supersede)
     (with-open-file (strl "help-messages-learn" :direction :output
                                     :if-does-not-exist :create :if-exists :supersede)
      (with-open-file (stra "help-messages-azimuth" :direction :output
                                       :if-does-not-exist :create :if-exists :supersede)
             (with-open-file (strp "help-messages-polar" :direction :output
                                         :if-does-not-exist :create :if-exists :supersede)
              (dolist (uni (unit.children 'ai3-kb-viewports 'member))
                (let ((mes (get.value uni 'help)))
                  (when mes
                        (let ((str (get-correct-stream uni strv strl stra strp)))
                          (princ uni str)
                          (princ " \"" str)
                          (princ mes str)
                          (princ "\" " str)))))
              (dolist (uni (unit.children 'ai3-unit-viewports 'member))
                (let ((mes (get.value uni 'help)))
                  (when mes
                        (let ((str (get-correct-stream uni strv strl stra strp)))
                          (princ uni str)
                          (princ " \"" str)
                          (princ mes str)
                          (princ "\" " str)))))
              (dolist (uni (unit.children 'ai3-slot-viewports 'member))
                (let ((mes (get.value uni 'help)))
                  (when mes
                        (let ((str (get-correct-stream uni strv strl stra strp)))
                          (princ uni str)
                          (princ " \"" str)
                          (princ mes str)
                          (princ "\" " str))))))))))

(defun get-correct-stream (uni strv strl stra strp)
"Returns the correct stream for the file holding the help messages for the
knowledge base containing the viewport."
  (case (unit.kbname uni)
    (VEG strv)
    (LEARN strl)
    (AZIMUTHPLOT stra)
    (POLARPLOT strp)))

(defun add-help ()
"Adds help for a viewport."
  (let ((unit (unit-from-stream (mouse-top-window))))
    (if unit
            (get-new-help unit)
            (my-documentation-print "Not a VEG window - help cannot be stored"))))
```

```
(defun get-new-help (unit)
"Prompts the user to enter the new help message and then accepts the new
message."
  (let ((old-mes (get.value unit 'help)))
    (put.value 'add.help 'unit unit)
    (cond ((or (null old-mes)
                 (equal old-mes "")(equal old-mes " ")(equal old-mes "  "))
            (remove.all.values 'add.help 'help.message)
            (put.value 'add.help 'message
"Enter the new help message.  Left click on DONE when finished.  <="))
           (t (put.value 'add.help 'help.message old-mes)
              (put.value 'add.help 'message
  "Modify the previous help message.  Left click on DONE when finished  <=")))
    (unitmsg 'viewport-add.help.4 'open-panel!)))

(defun make-one-long-string (list-of-strings)
"Concatenates a list of strings into one long string."
  (make-one-long-string-aux "" list-of-strings))

(defun make-one-long-string-aux (result remaining-strings)
 (if (null remaining-strings)
    result
    (make-one-long-string-aux (string-append result " "
                                               (first remaining-strings))
                               (rest remaining-strings))))

(defun wipe-out-help ()
"Removes all the help slots and help messages from all loaded knowledge bases."
   (delete.slot 'ai3-kb-viewports 'help)
   (remove.value 'ai3-kb-viewports 'local.compact.unit.slotnames 'help)
   (delete.slot 'ai3-unit-viewports 'help)
   (remove.value 'ai3-unit-viewports 'local.compact.unit.slotnames 'help)
   (delete.slot 'ai3-slot-viewports 'help)
   (remove.value 'ai3-slot-viewports 'local.compact.unit.slotnames 'help)
   (put.value 'help.system 'help.loaded nil))
```

Report Documentation Page

1. Report No.

2. Government Accession No.

3. Recipient's Catalog No.

4. Title and Subtitle

An Expert System Shell for Inferring Vegetation Characteristics - Prototype Help System (Task I)

5. Report Date

July 1993

6. Performing Organization Code

7. Author(s)

P. Ann Harrison and Patrick R. Harrison

8. Performing Organization Report No.

C931032-U-2R08

10. Work Unit No.

462-61-14

9. Performing Organization Name and Address

JJM Systems, Inc.
One Ivybrook Blvd., Suite 190
Ivyland, PA 18974

11. Contract or Grant No

NAS5-30127

12. Sponsoring Agency Name and Address

National Aeronautics and Space Administration
Washington, DC 20546-0001
NASA/Goddard Space Flight Center
Greenbelt, MD 20771

13. Type of Report and Period Covered

Task Report for Task I
April - July 1993

14. Sponsoring Agency Code

15. Supplementary Notes

The Lisp and KEE code for this work is available on a Sun Cartridge Tape.

16. Abstract

The NASA VEGetation Workbench (VEG) is a knowledge based system that infers vegetation characteristics from reflectance data. A prototype of the VEG subgoal HELP.SYSTEM has been completed and the Help System has been added to the VEG system. It is loaded when the user first clicks on the HELP.SYSTEM option in the Tool Box Menu. The Help System provides a user tool to support needed user information. It also provides interactive tools the scientist may use to develop new help messages and to modify existing help messages that are attached to VEG screens. The system automatically manages system and file operations needed to preserve new or modified help messages. The Help System was tested both as a help system development and a help system user tool.

17. Key Words (Suggested by Author(s))

EXPERT SYSTEM, ARTIFICIAL INTELLIGENCE, REMOTE SENSING

18. Distribution Statement

UNCLASSIFIED - UNLIMITED

19. Security Classif (of this report)

UNCLASSIFIED

20. Security Classif (of this page)

UNCLASSIFIED

21. No of pages

23

22. Price

For sale by the National Technical Information Service, Springfield, VA 22161-2171